# CS 5012: Binary Trees Module 4.3 Exercise

**H. Diana McSpadden (hdm5s)**

**3/2/2021**

## Question 1:

Assume the implementation of a BinaryTree from slide 30 in *Introduction to Tree*s. Draw the binary tree that the following code produces (the code was written in *main*).

# create a bunch of nodes with integer values

theRoot = BinaryTree.Node(3) # root node with value 3

n1 = BinaryTree.Node(1)

n2 = BinaryTree.Node(2)

n4 = BinaryTree.Node(4)

n5 = BinaryTree.Node(5)

n6 = BinaryTree.Node(6)

# create a binary tree called 'myTree'

myTree = BinaryTree(theRoot) # create a tree 'myTree' with root = 3

# connect the tree

myTree.root.setLeft(n1)

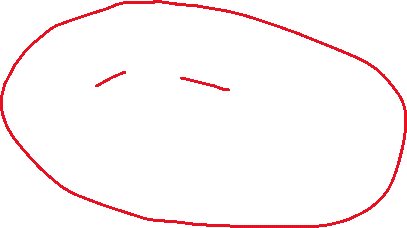
myTree.root.setRight(n4)

n1.setRight(n2)

n4.setRight(n6)



n4.setLeft(n5)



## Question 2:

Assume the implementation of a BinaryTree from slide 30 in *Introduction to Tree*s. Build a binary tree by providing the code in Python as would be typed in main (similar to the code in Q1), based on the following drawing.

**Answer**

# create the root node

theRoot2 = BinaryTree.Node(3)

# create the other nodes

nn1 = BinaryTree.Node(1)

nn2 = BinaryTree.Node(2)

nn11 = BinaryTree.Node(11)

nn5 = BinaryTree.Node(5)

nn6 = BinaryTree.Node(6)

nn9 = BinaryTree.Node(9)

# create the tree object

theTree2 = BinaryTree(theRoot2)

# connect the tree

theTree2.root.setLeft(nn1)

theTree2.root.setRight(nn6)

nn1.setLeft(nn2)

nn1.setRight(nn11)

n6.setRight(nn9)

nn6.setLeft(nn5)

## Question 3:

Write a tree-level *getHeight()* method that calculates the height (or depth) of the binary tree (e.g., it would return 3, if called on the tree in Q2).

Note: Think of a way to do this recursively, having each node calculate its height, as if it was the root of its subtree.

**Answer**

"""

Activity: MOE 4.3: Building a Binary Tree

Name: H. Diana McSpadden

UID: hdm5s

"""

class BinaryTree:

     # Methods of BinaryTree class below, including the constructor

    def \_\_init\_\_(self, root=None):

        self.root = root # identifying the root of the tree

    def getHeight(self, rootToUse):

        if rootToUse.right and rootToUse.left:

             # get height of left node

            leftHeight = self.getHeight(rootToUse.left)

            # get height of right node

            rightHeight = self.getHeight(rootToUse.right)

            if (leftHeight > rightHeight):

                return (1 + leftHeight)

            else:

                return (1 + rightHeight)

        elif rootToUse.left:

            return (1 + self.getHeight(rootToUse.left))

        elif rootToUse.right:

            return (1 + self.getHeight(rootToUse.right))

        else:

            # height is 1 at a leaf, i.e. no children

            return 1

    '''

    Internal Class Node

    Used to hold properties of each node

    '''

    class Node: # Node class, internal to BinaryTree

        def \_\_init\_\_(self, val, left=None, right=None):

            self.val = val # data item

            self.left = left # left child

            self.right = right # right child

        def getVal(self):

            return self.val

        def setVal(self,newval):

            self.val = newval

        def setLeft(self,newleft):

            self.left = newleft

        def setRight(self,newright):

            self.right = newright

        def getLeft(self):

            return self.left

        def getRight(self):

            return self.right

        ## -------------- end of internal class Node --------------

# create a bunch of nodes with integer values

theRoot = BinaryTree.Node(3) # root node with value 3

n1 = BinaryTree.Node(1)

n2 = BinaryTree.Node(2)

n4 = BinaryTree.Node(4)

n5 = BinaryTree.Node(5)

n6 = BinaryTree.Node(6)

# create a binary tree called 'myTree'

myTree = BinaryTree(theRoot) # create a tree 'myTree' with root = 3

# connect the tree

myTree.root.setLeft(n1)

myTree.root.setRight(n4)

n1.setRight(n2)

n4.setRight(n6)

n4.setLeft(n5)

height = myTree.getHeight(theRoot)

print(height)

## References:

Joe James: https://www.youtube.com/watch?v=aGaMgkJX5-o